[*Proceedings of the National Symposium on Machine Translation*, UCLA February 1960]

Session 8: INFORMATION PROCESSING AND LINGUISTIC ANALYSIS

A NEW THEORY OF TRANSLATION AND ITS APPLICATIONS[1]

Anthony G. Oettinger
Harvard University

The technique of predictive analysis and translation assumes a particularly simple limiting form [1] with respect to certain artificial languages, of which the following are examples:

(1) L: The Łukasiewicz parenthesis-free notation [2]. In the following, x_i denotes a variable, δ_{jk}, the k^{th} member of a set of functors of degree j, and Δ^j_i ; an arbitrary well-formed formula in L_j .

(2) L_1 : A language in which the well-formed formulas are:

(a) x_i , and

(b) if Δ^1_1 and Δ^1_2 , then $(\delta_{1j}\ \Delta^1_1$, and also $(\Delta^1_1\ \delta_{2j}\ \Delta^1_2$.

(3) L_2 : A language in which the well-formed formulas are:

(a) x_i , and

(b) if Δ^2_1 and Δ^2_2 , then $\delta_{1j}\ \Delta^2_1)$, and also $\Delta^2_1\ \delta_{2k}\ \Delta^2_2)$.

(4) L_3: A language in which the well-formed formulas are:

(a) x_i , and

(b) if Δ^3_1 and Δ^3_2 , then $(\delta_{1j}\ \Delta^3_1)$, and also $(\Delta^3_1\ \delta_{2k}\ \Delta^3_2)$.

L_1, L_2, and L_3 will be referred to respectively as left-parenthetic, right-parenthetic, and simple full-parenthetic languages.

Let p be a pushdown store. Let the input formula be scanned character-by-character from left to right, and let the output formula be produced by adjoining each new character to the left of those previously generated. Let every functor δ_{jk} of L_i have an image δ'_{jk} in L as, for example, $\sim\ <\rightarrow N$, $+\ <\rightarrow A$, $\bullet \leftrightarrow M$. With these conventions, rules for translating from L_2 to L may be given as follows:

If the current input character is

(1) a functor, put its image at the top of p ;

(2) a variable, transfer it to the output;

---

[1] This work has been supported in part by the National Science Foundation and by the Rome Air Development Center, Air Research and Development Command.

(3) a right parenthesis, transfer the character currently at the top of p to the output, then remove it from p .

The rules for translating from L_1 to L are only slightly more complex:

If the current input character is

(1) a left parenthesis, put a "v" at the top of p ;

(2) a functor, replace the "v" at the top of p by the image of the functor;

(3) a variable

(a) transfer it to the output;

(b) check p : if it is empty or has a "v" on top, proceed to the next input character; otherwise transfer the character currently at the top of p to the output, then remove it from p , and repeat step (b).

These algorithms, as well as their inverses, and algorithms for translating in either direction between any pair of members of $\{L, L_1, L_2, L_3, \dots\}$ can easily be described in a new notation recently devised by Iverson [3] which lends itself well to the formulation of proofs of certain interesting and significant properties of the algorithms.

For example, algorithms for translating from L to L_3 and vice-versa have been devised for which it can be proved that they will produce an image formula if and only if the input formula is well-formed in the domain of translation. The image in each case is unique, and well-formed in the range.

Let $\Delta = \Delta_H \; \Delta_M \; \Delta_T$ be any formula of the domain, split into a head Δ_H , a middle Δ_M , and a tail Δ_T . Δ_M is well-formed in the domain, while Δ_H and Δ_T are arbitrary residues determined by the choice of Δ_M . At a certain point in the execution of an algorithm, the remaining input formula will be $\Delta_M \; \Delta_T$, some image Δ'_H of Δ_H will have been previously generated, and p will be $p(\Delta_H)$, namely a function of Δ_H only. While the characters of Δ_M are being scanned, p naturally becomes a function of Δ_M as well as of Δ_H , but all contributions to p due to Δ_M will be "above" those due to Δ_H in the pushdown store.

Every algorithm of the type under consideration operating on formulas of the kind described in the preceding paragraph obeys the conditions of a Δ_M-theorem which guarantees, for any well-formed, Δ_M that once the remaining input formula is Δ_T, then

(1) p is again $p(\Delta_H)$, that is, no contributions due to Δ_M remain, at the top of the pushdown store,

(2) the well-formed image Δ'_M of ΔM will have been adjoined to Δ'_H.

By way of illustrating the implications of this theorem we note that an algorithm obeying it treats any nested well-formed subformula independently of the rest of the formula. As a consequence, such algorithms, if fail-safe, are fail-safe in a particularly satisfactory way: as one example, taken from natural languages, prepositional phrases or subordinate clauses can emerge unscathed, even though the sentence in which they are embedded may not be analyzable as a whole; as another example, from automatic programming, all the well-formed subroutines of a program could be found at a single pass through a compiler, even though the program as a whole might not be well-formed. Debugging could therefore be made considerably easier than it is in contemporary practice. Metaphorically speaking, any branch of a tree can be analyzed even though it has been broken off its parent branch.

A more complete and detailed description of these results, including proofs of the relevant theorems, is being prepared for publication.

REFERENCES

[1] Oettinger, A. G. , "Current Research on Automatic Translation at Harvard University", paper presented at the National Symposium on Machine Translation, Los Angeles (1960).

[2] Burks, A. W. , Warren, D. W. , and Wright, J.B., "An Analysis of a Logical Machine Using Parenthesis-free Notation", MTAC, Vol. VIII, No. 46, pp. 53-57 (1954).

[3] Iverson, K. E. , "The Description of Finite Sequential Processes", Theory of Switching, Report No. BL-23, Section III, Harvard Computation Laboratory, Cambridge, Massachusetts.